

Chapter 0

Exercises

Course “Compiler Construction”

Martin Steffen

Spring 2022



Section

Exercises 08

8.1: Code generation (repetition)
2. Code generation (2007, 4a): liveness
3. Code generation (2009,4): BB's, CFG
8.4: Code generation (2010, 4d)
Code generation (2011 4c,4d)

Chapter 0 "Exercises"
Course "Compiler Construction"
Martin Steffen
Spring 2022



Exercises 08

- 8.1: Code generation (repetition)
- 2. Code generation (2007, 4a): liveness
- 3. Code generation (2009,4): BB's, CFG
- 8.4: Code generation (2010, 4d)
- Code generation (2011 4c,4d)

Exercises 08

8.1: Code generation (repetition)

2. Code generation (2007, 4a): liveness

3. Code generation (2009,4): BB's, CFG

8.4: Code generation (2010, 4d)

Code generation (2011 4c,4d)

8.1.1 + 8.1.2

- re-cap of the code generator?
- possible improvements?



8.1.3: 3AIC

```
1  t := a - c
2  u := a + c
3  w := a / t
4  d := w + u
```

Listing: 3AIC

8.2: Liveness in 3AIC “source” code (exam 2007)

```
1  a := input
2  b := input
3  d := a + b
4  c := a * b    // <- looky here
5  if ( b < 5) {
6    while (b < 0 ) {
7      a := b + 2
8      b := b + 1
9    }
10   d := 2 * b
11 } else {
12   d := b * 3
13   a := d - b
14 }
15 output a
16 output b
```



Exercises 08

8.1: Code generation (repetition)

2. Code generation (2007, 4a): liveness

3. Code generation (2009,4): BB's, CFG

8.4: Code generation (2010, 4d)

Code generation (2011 4c,4d)

8.3: Code generation (exam 2009. 4)

```
1  a := input
2  b := input
3  t1 := a + b   // line 3
4  t2 := a * 2
5  c := t1 +  t2
6  if a < c goto 8
7  t2 := a + b
8  b := 25       // line 8
9  c := b + c
10 d := a - b
11 if t2 = 0 goto 17
12 d := a + b
13 t1 := b - c
14 c := d - t1
15 if c < d goto 3
16 c := a + b
17 output c      // line 17
18 output d
```

Listing: 3-address code example



Exercises 08

8.1: Code generation (repetition)

2. Code generation (2007, 4a): liveness

3. Code generation (2009,4): BB's, CFG

8.4: Code generation (2010, 4d)

Code generation (2011 4c,4d)

8.4: Code generation: 3AIC to 2AC (2010)

```
1  t1 := a - b
2  t2 := b - c
```

Listing: source

Is the following code generated from the 3AIC?

```
1  MOV  a, R0
2  MOV  b, R1
3  SUB R1, R0
4  SUB  c, R1
```

Listing: generated 2AC?



Exercises 08

8.1: Code generation (repetition)

2. Code generation (2007, 4a): liveness

3. Code generation (2009,4): BB's, CFG

8.4: Code generation (2010, 4d)

Code generation (2011 4c,4d)

8.5: P-code (2011)

`lda v`	"load address"	Determine the address of variable v and push it on top of the stack. An address is an integer number, as well.
`ldv v`	"load value"	Fetch the value of variable v and push it on top of the stack
`ldc k`	"load constant"	Push the constant value k on top of the stack
`add`	"addition"	calculate the sum of the stack's top two elements, remove ("pop") both from the stack and push the result onto the top of the stack.
`sto`	"store"	
`jmp L`	"jump"	goto the designated label
`jge L`	"jump on greater-or-equal"	similar conditional jumps ("greater-than", "less-than" ...) exist.
`lab L`	"label"	label to be used as targets for (conditional) jumps.

Table: P-code instructions (for reference)